

String Vector based KNN and AHC Algorithm for Automatic Text Summarization based on Text Clustering

Taeho Jo
President
Alpha AI Research
Cheongju, South Korea
tjo018@naver.com

Abstract—In this research, we propose the string vector based KNN algorithm and the string vector based AHC algorithm for automating the text summarization, based on the text clustering. In the previous work, even if the KNN version was applied to the text summarization, a domain should be presented as an input text tag. In this research, text subgroups are constructed based on their contents, by the text clustering, and each paragraph from a text is classified by selecting a classifier which corresponds to its most similar cluster. The AHC algorithm and the KNN algorithm which process string vectors are adopted respectively as the approaches to the text clustering and the text summarization. The goals of this research are to automate the text summarization and to improve the performance of both tasks, using the string vector-based algorithms.

I. INTRODUCTION

The text summarization is defined as the process of extracting the essential part from a text as its summary. It is necessary to judge whether each text partition is essential, or not. A text partition is determined as a paragraph, and the text summarization is interpreted into the classification of each paragraph into summary or non-summary. Paragraphs which are classified into summary are extracted as the summary of the text. This research is intended to apply the better machine learning algorithm to the classification task which is mapped from the text summarization.

This research is motivated by the necessity of defining domains manually in designing the text summarization system. The text summarization is mapped into a binary classification which is dependent on a domain; the text summarization system is decomposed into binary classification systems as many as domains. The process of designing the text summarization system is to predefined domains depending on prior knowledge and subjective views, collect sample paragraphs domain by domain, and allocate a binary classifier for each domain. One more motivation of this research is the discovery of the excellent performances of the string vector based KNN algorithm and the string vector based AHC algorithm in the text summarization and the text clustering. This research is intended to automate the domain predefinition by connecting the text summarization with the text clustering.

In this research, we propose that the text summarization should relate to the text clustering, and the string vector based KNN algorithm and the string vector based AHC algorithm should be applied to both tasks. In this research, we initially prepare the text collection where each text is associated with its own summary. The semantic similarity metric between two string vectors is defined, and the KNN algorithm and the AHC algorithm are modified based on the metric into the string vector-based versions. The role of text clustering is to segment the text collection into subgroups each of which consists of content based similar texts, and a binary classifier which performs the text summarization is implemented within each subgroup independently. Each subgroup corresponds to a domain, and it is decided to an input text by its similarities with the cluster prototypes automatically.

Let us mention some benefits from this research. The main problems in encoding texts into numerical vectors are overcome by encoding them into string vectors. The performances in the text summarization and the text clustering are improved by adopting the string vector-based versions of the KNN algorithm and the AHC algorithm as the approaches to both tasks. We avoid defining the domains depending on prior knowledge or subjective views by automating the domain predefinition through the text clustering. We avoid deciding a domain in advance in summarizing a text by computing the similarities of an input text with the cluster prototypes.

Let us mention some remaining tasks as the further discussions on this research. We should modify more advanced machine learning algorithms and deep learning algorithms into string vector-based versions. We need to define and characterize mathematically additional operations on string vectors. We need to implement the text summarization system which relates to the text clustering as a real program or a library. The text classification and the text clustering may relate with the text summarization for improving their performances.

II. PROPOSED WORKS

A. Encoding Proccers

This section is concerned with the string vector as the text representation, and the similarity between string vectors. A string vector is a finite ordered set of strings; numerical values are replaced by strings as the elements in a vector. It is required to define the similarity matrix for performing the operation on string vectors. The similarity between string vectors is the average over one-to-one similarities between their elements. This section is intended to describe the process of encoding a text into a string and the process of computing the similarity between string vectors.

The process of encoding a text into a string vector is illustrated in Figure 1. A text is indexed into a list of words and their information. The features of a string vector are defined as symbolic forms manually in the feature definition. A string which represents a text is filled with the words which corresponds to the features in the feature value assignment. Refer to [2] for studying the encoding process in more detail.

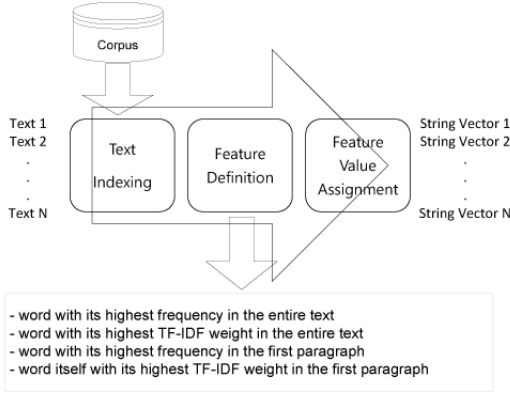


Figure 1. Process of Encoding Texts into String Vectors

The similarity matrix which is the basis for computing the similarity between two string vectors is illustrated in Figure 2. Each row and each column correspond to their own word, and each element in the similarity matrix is the similarity between two words, t_1 and t_2 , as shown in equation (1),

$$s_{ij} = \text{sim}(t_i, t_j) \quad (1)$$

The similarity between two words, t_i and t_j is computed by equation (2),

$$\text{sim}(d_i, d_j) = \frac{2\#(t_i \wedge t_j)}{\#(t_i) + \#(t_j)} \quad (2)$$

where $\#(t_i \wedge t_j)$ is the number of texts which include both t_i and t_j , $\#(t_i)$ is the number of texts which includes t_i , and $\#(t_j)$ is the number of texts which includes t_j . The value of similarity between two words, t_i and t_j , is always given

as a normalized value between zero and one, as shown in equation (3),

$$0 \leq \text{sim}(t_i, t_j) \leq 1 \quad (3)$$

The similarity matrix is used for computing the similarity between string vectors which represent texts as the reference table.

$$\begin{matrix} & \text{Word 1} & \text{Word 2} & \dots & \text{Word N} \\ \text{Word 1} & S_{11} & S_{12} & \dots & S_{1N} \\ \text{Word 2} & S_{21} & S_{22} & \dots & S_{2N} \\ \dots & \dots & \dots & \dots & \dots \\ \text{Word N} & S_{N1} & S_{N2} & \dots & S_{NN} \end{matrix} \quad s_{ij} = \frac{2 \times \#(\text{word}_i, \text{word}_j)}{\#(\text{word}_i) + \#(\text{word}_j)}$$

Figure 2. Semantic Similarity Matrix

Let us mention the process of computing the similarity between string vectors as a semantic similarity between texts. The two string vectors which represent texts are notated by $\text{str}_1 = [t_{11} \ t_{12} \ \dots \ t_{1d}]$ and $\text{str}_2 = [t_{21} \ t_{22} \ \dots \ t_{2d}]$. The similarity between two string vectors is computed by equation (4),

$$\text{sim}(\text{str}_1, \text{str}_2) = \frac{1}{d} \sum_{i=1}^d \text{sim}(t_{1i}, t_{2i}). \quad (4)$$

The similarity between elements, $\text{sim}(t_{1i}, t_{2i})$, is taken from the similarity which was mentioned above. The value which is generated by equation (4) is always given as a normalized value between zero and one as shown in equation (5),

$$0 \leq \text{sim}(\text{str}_1, \text{str}_2) \leq 1. \quad (5)$$

Let us make some remarks on the encoding process and the computation of the similarity between texts. A text is encoded into a string vector by the word-text association, the feature definition, and the feature value assignment. The similarity matrix is defined as the basis for computing the similarity between two string vectors. The similarity between two string vectors is computed by equation (4). In the future research, we consider representing a text into multiple string vectors.

B. String Vector based KNN Algorithm

This section is concerned with the string vector based KNN algorithm. It was initially proposed as the approach to the text summarization by Jo in 2018 [1]. The similarity metric between string vectors which was described in the previous section is used for computing the similarity between a novice string vector and a training example. The labels of its nearest neighbors are voted with their identical weights for deciding the label of a novice input. This section

is intended to describe the initial version of KNN algorithm from which its variants are derived.

The process of computing the similarity between a novice item and a training example is illustrated in Figure 00. The labeled words which are gathered as samples are encoded into string vectors, and they are notated by a set, $Tr = \{\mathbf{str}_1, \mathbf{str}_2, \dots, \mathbf{str}_N\}$. A novice word is encoded into a string vector notated by $\mathbf{str}$. Its similarities with the string vectors which represent the sample words are computed by equation (??), as $sim(\mathbf{str}, \mathbf{str}_1), sim(\mathbf{str}, \mathbf{str}_2), \dots, sim(\mathbf{str}, \mathbf{str}_N)$. Some training examples which are most similar as the novice input are selected as its nearest neighbors.

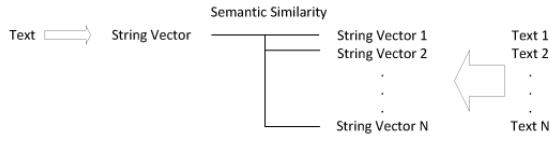


Figure 3. Similarities of Novice Input with Training Examples

In Figure 4, the process of selecting nearest neighbors by the ranking selection is illustrated. The training examples are sorted by the descending order of their similarities, after computing their similarities with a novice example. The training examples with their higher similarities with a novice example are selected as its nearest neighbors, as expressed in equation (6),

$$Nr = Select_k(Tr, \mathbf{str}), Nr \subseteq Tr \quad (6)$$

The set of nearest neighbors, Nr , is composed with elements as expressed in equation (7),

$$Nr = \{\mathbf{str}_1^{near}, \mathbf{str}_2^{near}, \dots, \mathbf{str}_k^{near}\} \quad (7)$$

The elements in the set, Nr , are used for voting their labels in the next step.

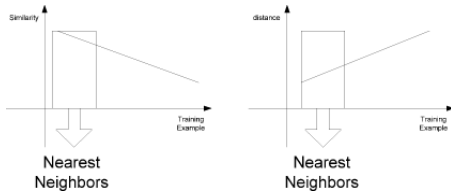


Figure 4. Selection of Most Similar or Least Distant Training Examples as Nearest Neighbors

The process of voting the labels of nearest neighbors for decoding the label of a novice word is illustrated in Figure 5. The predefined categories are notated by $c_1, c_2, \dots, c_m$, and the label of a nearest neighbor is notated by $c_j = label(\mathbf{str}_i^{near})$. The number of nearest neighbors which belong to the category, c_j , is notated by $Count(Nr, c_j)$.

The label of the novice item, $\mathbf{str}$, is decided by equation (8),

$$label(\mathbf{str}) = \underset{j=1}{\operatorname{argmax}}^m Count(Nr, c_j) \quad (8)$$

The process of deciding the label of a novice item by equation (8), is called voting.



Figure 5. Voting of Labels of Nearest Neighbors for Deciding Label of Novice Input

Let us make some remarks on the string vector based KNN algorithm. It computes the similarities of a novice item with the training examples. The training examples with their highest similarities with the novice item are selected as its nearest neighbors. The label of the novice item is decided by voting the labels of its nearest neighbors. The ranking selection is adopted for selecting the nearest neighbors from training examples, in this version.

C. System Architecture

This section is concerned with the text summarization system which adopts the string vector based KNN algorithm. In Section II-B, we described the proposed version of KNN algorithm as the approach to the text summarization. It is viewed as a binary classification which classifies a paragraph into summary or non-summary. This section is intended to describe the text summarization system with respect to its function and its architecture.

The process of sampling paragraphs domain by domain is illustrated in Figure 6. Even if it is possible map the text summarization into an instance of text classification, a paragraph may be classified differently depending on the domain. Sample paragraphs which are labeled with summary or non-summary are gathered domain by domain. The text which is given as the input is tagged with a domain, and the classifier which corresponds to the domain is appointed. The sample paragraphs are encoded into string vectors in each domain.

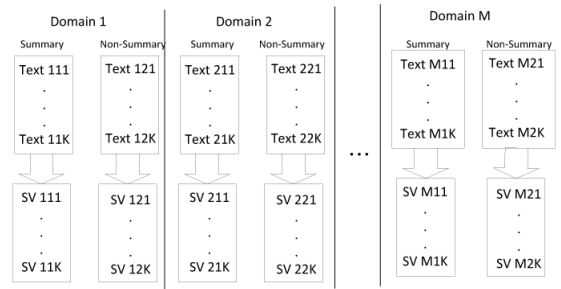


Figure 6. Preparation of Training Examples

The entire architecture of the proposed text summarization system is illustrated in Figure 7. A text is given as the input, and paragraphs are extracted by partitioning the text. In the encoding module, the sample paragraphs in the summary group and the non-summary group and ones which are extracted from the text are mapped into string vectors in the encoding module. The paragraphs from the text are classified into one of the two categories in the similarity computation module and the voting module. The paragraphs which are classified into summary are extracted as the output in this system.

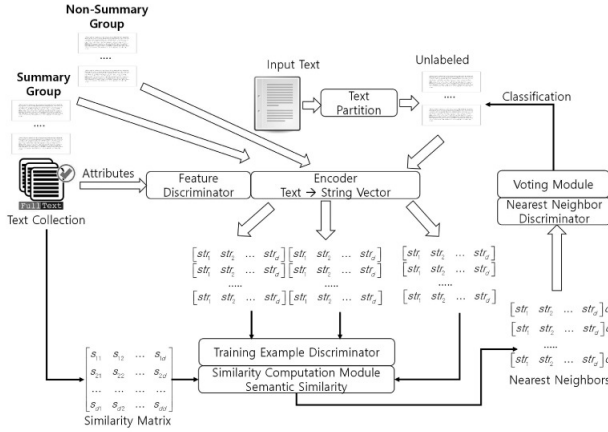


Figure 7. System Architecture

The execution flow of the text summarization system is illustrated in Figure 8. In each domain, sample paragraphs which are labeled with summary or non-summary are gathered. The input text is partitioned into novice paragraphs, and both sample paragraphs and novice paragraphs are encoded into string vectors. Each paragraph is classified into summary or non-summary, and there are two groups of paragraphs in the text: summary group and non-summary group. The paragraph which are classified into summary are extracted as the summary of the input text in this system.

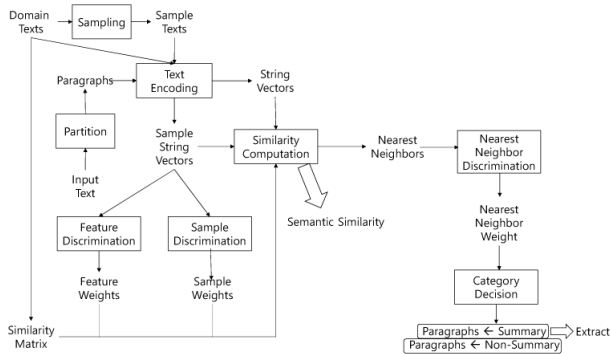


Figure 8. System Flow

Let us make some remarks on the proposed system

which is illustrated in Figure 7 as its architecture. The text summarization is defined as the binary classification where each paragraph is classified into summary or non-summary. Paragraphs are encoded into string vectors instead of numerical vectors, and they are classified directly. Ones which are classified with summary are selected as the summary of the input text. We need to add the text classification module for predicting the domain of input text automatically.

D. Connection with Text Clustering

This section is concerned with the connection of the text summarization with the text clustering. In the previous section, we mentioned the text summarization as an instance of domain dependent classification. The domains are automatically constructed from the text collection by clustering texts. The AHC algorithm which is proposed in [2] is used for implementing the text clustering. This section is intended to describe the text summarization system which is connected with the text clustering for automating the construction of domains.

The architecture of the text clustering system which was proposed in [2] is illustrated in Figure 9. The texts in the group are encoded into string vectors, and the AHC algorithm which is proposed in [2] is adopted as the approach. It starts with singletons as many as items, computes the similarities of all possible cluster pairs, and merge the cluster pair with its highest similarity into one cluster. The two steps are iterated until the number of clusters reach the desired number. We may consider replacing the AHC algorithm by its variants for improving the speed and the performance.

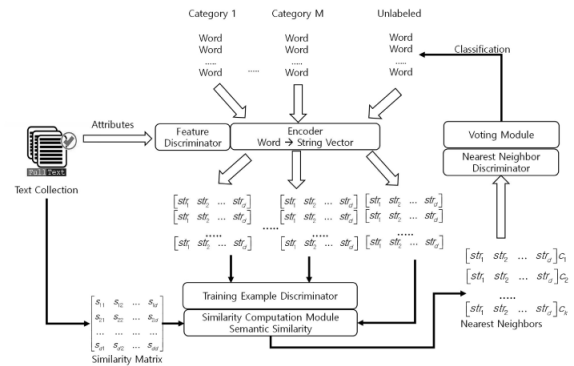


Figure 9. Text Clustering System

The connection of the text summarization with the text clustering is illustrated in Figure 10. The M domains are defined by clustering texts in a group, initially and automatically. A novice text is given as the input, and its domain, domain D , is decided by its similarities with domains. A classifier which corresponds to domain D is nominated and classify each paragraph in a novice text into summary or non-summary. The M text summarization systems are

viewed as independent ones, each of which classifies each paragraph into one of the two categories.

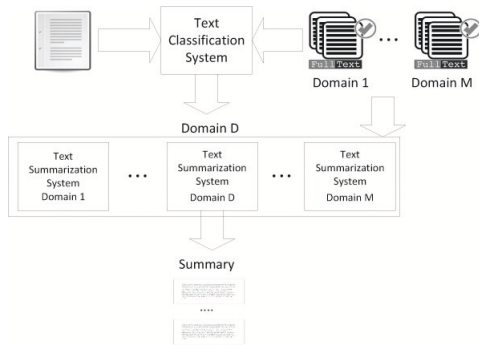


Figure 10. Text Summarization System connected with Text Clustering

The execution flow of text summarization which is connected with the text clustering is illustrated in Figure 11. Unlabeled texts are clustered into subgroups, each of which is a domain. Sample paragraphs which are labeled with summary or non-summary are generated from each domain. These sample paragraphs are provided for training the classifier in each text summarization system. Each classifier is used for judging whether each paragraph in the input text is essential or not.

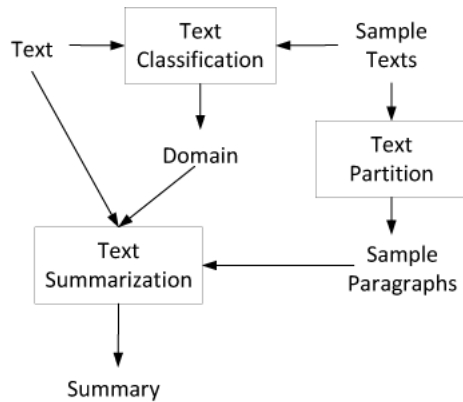


Figure 11. System Flow of Text Summarization connected with Text Clustering

Let us make some remarks on the connection of the text summarization with the text clustering. It is the process of segmenting a text group into subgroups based on their content-based similarities. The role of text clustering is to define the domains initially in the architecture of connecting the text summarization with the text clustering. In each domain, sample paragraphs are generated, and its corresponding classifier is trained with them. In the future research, we consider using the text summarization for clustering texts in the opposite direction.

REFERENCES

- [1] T. Jo, "Automatic Text Summarization using String Vector based K Nearest Neighbor", 6005-6016, Journal of Intelligent and Fuzzy Systems, 35, 2018.
- [2] T. Jo, "Semantic String Operation for Specializing AHC Algorithm for Text Clustering", 1083-1100, Annals of Mathematics and Artificial Intelligence, 2019.